MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SERVOMECHANISMS LABORATORY
Cambridge 39, Massachusetts

M E M O R A N D U M

SUBJECT: MOUSE--Preliminary Instructions

FROM: John E. Ward

DATE: January 16, 1959

---

1. General Description

The MOUSE Program was written primarily as an exercise in programming to learn the features and capabilities of the TX-0. It was also written with the idea of having a good demonstration program for visitors, and considerable effort therefore went into the displays.

The program permits one to generate or alter a maze and insert or move a mouse and up to three pieces of cheese by means of the light pen under TAC control. Operation of the program and control of the mouse is also by means of TAC. The program tape includes UT-3 for convenience in working with the program, making special entries, and obtaining punchout of mazes for later use. A number of sample mazes have been included at the end of the tape and can be brought into the computer by successive read-ins and restarts.

The program occupies substantially all the storage in the computer between registers 20 and 4707. Of this about 1000 registers are devoted to data storage for the various displays and about 3600 registers to working program. Program cycle time depends on the number of walls and is about 1/2 sec. for typical problems. The program contains a few bugs, as described in Section 6 , which may be fixed sometime.

D.T. Ross wrote the part of the program which involves the mouse's logic in solving the maze. Displays, TAC and light pen control, and other features were written by J.E. Ward.

2. Read-In and Starting Instructions

To operate the program, the display switch must be on, the flexowriter must be turned on and primed by means of its start read switch (the program types out comments on the flexowriter under some

circumstances), and the type-on switch must be on.

Place the tape in the PETR and push the read-in button. After the tape stops, a restart will cause an undeveloped maze with all walls in place to appear on the display scope. Further operations with this display are described in Section 3.

If one of the sample mazes is desired rather than the "clean slate" display described above, push the read-in button again instead of the restart. Upon restart after this second read-in, a sample maze will appear which contains a mouse and three pieces of cheese. Here again, initiation of the program from this point is controlled by TAC as described in Section 3. The remaining three sample mazes can be brought into the computer and initiated by the same process of read-in and restart.

The main starting addresses for various modes of operation are as follows.

(a) Start at 100 erases the wall, mouse, and cheese storage in the program and new data must be inserted by means of the light pen.

(b) Start at 2371 retains the wall, mouse, and cheese storage but erases the mouse's memory and puts the program in a start over mode.

(c) Start at 4400 is the same as a start at 2371 except that the mouse's eating habits are changed (for VIP visitors only).

(d) Start at 4545 after (c) restores the program to normal operation.

3. Operation with TAC and Light Pen

a. TAC Language

A language has been established for the switches in TAC. Bit zero has the meaning "do," bit 13 means "write," bit 14 means "erase," bit 15 means "wall," bit 16 means "mouse" and bit 17 means "cheese." Switches 1-12 are ignored. A TAC statement must include "do" plus a legal combination of the other five switches, otherwise, the program response is "WHAT ?" displayed on the scope. Legal combinations of the switches are as follows.

do write walls	(400024)
do erase walls	(400014)

do write mouse	(400022)
do erase mouse	(400012)
do write cheese	(400021)
do erase cheese	(400011)
do mouse	(400002)
do over	(400017)
do erase storage	(400037)

The last two combinations do not follow any language rules but are a convenient means for starting the program over again at registers 2371 and 100 respectively without stopping the program.

For convenience, a language card has been made up which fits under TAC. The card is mounted by slipping its end under the panel moulding at the right of TAC, and aligning "do" with bit 0. The card should be kept with the tape when not in use.

b. Setting Up a Problem

In order to set up or alter any maze problem, insert the appropriate statement in TAC and use the light pen. In erasing walls, be careful to hold the pen over the center of the wall being erased, otherwise, adjacent walls may disappear as well. However, walls can be put back. In the "write wall" mode, a dot is displayed in the center of each space where a wall has been erased. Holding the pen over this point will cause the erased wall to reappear.

In the "write mouse" and "write cheese" modes, a raster of dots is displayed in the center of each box and a mouse or a cheese will be written in when a pen is held over these dots. To erase a mouse or cheese, hold the pen over the mouse or cheese display.

The program has provisions for insertion of three pieces of cheese and will not accept more than three, The program also contains interlocks to prevent writing more than one cheese in a given box, with the exception that any cheese which has been eaten by the mouse can be written over. Note that in writing a cheese the program will store a new cheese in the first vacant storage space or in the first storage space occupied by a cheese which has been eaten. Thus, any cheese which has been eaten may be moved to a new location during this process.

c. Control of the Mouse

Once the maze has been set up, the mouse may be started by the statement "do mouse." The mouse is a clever fellow and will not start unless at least one cheese is present in the maze. The mouse will stop and eat a cheese when he finds it and will then either (a) go on if there are other cheeses in the maze, or (b) stop if he has eaten the last cheese. The three cheese spots which remain are crumbs. After the

mouse has stopped, he can be asked to rerun the maze, retaining his memory which was built up during the previous run. This mode is obtained by saying "do mouse" again. In case the "do mouse" statement is still in TAC from the previous start, the mouse requires that you say it again by turning the "do" switch off and on again. In case a rerun with memory is not desired, say "do over" and the mouse will be taken back to the starting point and his memory erased, after which "do mouse" will cause him to solve the problem all over again.

One feature of the mouse is that he has an energy limit and will become pooped and stop if he has not found a cheese before his energy runs out. When this happens, he may either be fed a cheese by means of the light pen and the statement "do write cheese," or he can be asked to rerun with memory by saying "do mouse" again. If he is fed, he will eat the cheese (after the "do" switch is turned off) and start off with new energy and look for more cheese. If he is asked to rerun when pooped, he will of course retain his memory of the maze as far as he had gotten the first time and will be able to go much farther than the first time and hopefully find the cheese. Two different energy levels are used, depending on how many pieces of cheese were put in the maze at the start. If only one cheese was written, the mouse starts with an energy of 300 trials or moves and the same amount is added to his remaining energy when he eats a cheese. If two or more pieces of cheese were in the maze at the start, his initial energy and the increment he obtains when he eats each cheese is 100 trials and moves. If one is interested in seeing how efficiently the mouse solves various mazes on initial runs and reruns, his accrued energy when he stops is available for inspection in register 3116, stored as a negative value. The reset value for low energy (-100) is in register 2170, and the value for high energy (-300) is in register 2171. Changing register 2171 to a large negative number, say -4000, before a one-cheese run would prevent the mouse from ever becoming pooped.

If any walls are written or erased after a run, the mouse's memory is erased and he cannot be asked to rerun with memory. A "do mouse" after changing any walls will cause the mouse to consider the maze a new problem to solve.

If one is inconsiderate and places the cheese in an inaccessible location, or places the mouse in a box with no exits, the mouse will explore the entire space available to him before stopping and typing out an appropriate comment on the typewriter. He cannot be moved from this position by "do mouse" unless he is fed or unless a wall is written or erased somewhere in the maze. Also, he has unfortunately become so discouraged that he cannot smell a new cheese inserted in the space accessible to him unless it is put in the same box with him, i.e., unless he is fed.

4. Permanent Maze Records

If in operation of the program a particularly interesting maze is developed and it is desired to save it for later use, the procedure is as follows. Stop the program in the "WHAT?" loop by making any illegal TAC statement and transfer to UT-3 (5600) by means of the Test mode and the TBR. Type input, followed by punch 400 to 432, and start at 2371. This tape will contain the data storage for the walls, the cheese positions, and the initial position for the mouse. It is not necessary to return the mouse to the initial position before the punch-out because the display position for the mouse is in a different register than the initial position. After punching out the maze tape, the program can be resumed by typing "go to 1500."

5. Mouse Behavior

No attempt will be made here to describe the mouse completely. However, it may be well to describe the mouse's logic briefly.

The mouse attempts to go straight ahead as far as possible and will only look to the right or left if he finds a wall in front of him. As he proceeds, he makes a "path" mark in his wall memory to indicate each wall that he tries and finds open. If he later reverses this same path, he changes the mark to one which indicates that the path is blind and therefore should not be entered again. Upon a rerun, this "blind alley" mark has the same effect as an actual wall. The path mark is useful because it prevents the mouse from looping endlessly around an unconnected wall. As soon as he stumbles on his previous path, as indicated by the path mark, he will realize that he has looped and back up to the next untried path.

The mouse's present logic is fairly efficient when he is going forward but is less efficient when he is backing up, and he therefore appears to be overly cautious when backing up. Subsequent program changes may improve his backup logic, but these are not planned for the time being.

6. Cautions and Reservations

The present version of the program has a few undiscovered flaws and under some circumstances the mouse has fourth-dimensional tendencies which cause him to blithely ignore walls and the topographical problem of getting from one edge of the maze to the other. The only suggestion if this tendency develops is to erase his memory by "do over," or make another maze. No guarantees are given as to when (if ever) this type of behavior in the program will be corrected.

Complete control of the program is provided by TAC, and it is usually not necessary to push the stop button if something undesired is

happening. All that is necessary is to say "do" and the problem will cycle in the "WHAT?" loop. It is also important not to leave the program to go to UT-3 unless the program is cycling in the "WHAT?" loop, otherwise, the maze will become completely garbled. This is because the maze is generated by cycling storage words and if the computer is stopped during this process, the maze storage in the program is permanently altered. Return from UT-3 should be made to register 1500 if it is desired to continue the same problem. Return to 100, 2371, 4400, or 4545 will have actions as described in Section 2.

One final caution is that the mouse has no experience in the world outside his maze, and outer walls of the maze should not be erased, otherwise, he will escape into the fourth dimension. When this occurs, various and sometimes hilarious behavior occurs, but certainly nothing intended in the design of the program.